RETDAT Timing

Apr 18, 2000

Measurements are made from task timing diagnostics, with all times given in ms. The meanings of the tasks are:

SNAP	Performs IP network support upon datagram arrival
ANet	Acnet task dispatches message based upon task name or task-id
ACReq	RETDAT processing, either for requests or replies
Updt	Update task processing builds initial reply
Total	Sum of timing for SNAP, ANet, ACReq, Updt
ΔUpdt	Approximate time for Update task to build reply
ΔServ	Approximate time for Server task to build reply

Nonserver request timing

The following statistics measure the time for an IRM to handle a **nonserver** request. Several sizes of requests are used, with varying number of devices. Each one consists of a request for a 2-byte reading and a 2-byte setting for each device, or two PI/DI's per device. The requests are for 7.5Hz replies. This allows measuring the time for readings averaging that occurs every cycle, whether the reply is to be delivered or not.

#dev	*SNAP*	*ANet*	*ACReq*	*Updt*	*Total*	*ΔUpdt*
1	0.19	0.10	0.35	0.42	1.06	0.15–0.3
10	0.22	0.10	0.81	0.62	1.75	0.2–0.4
20	0.26	0.09	1.31	0.80	2.46	0.4–0.9
40	0.34	0.10	2.35	1.19	3.98	0.1–1.0
80	1.10	0.10	4.45	1.98	7.63	0.3–2.0

The time needed to provide IP support depends upon the size of the message, since the UDP checksum must be checked. The times needed by the Acnet task are constant, since Acnet only does dispatching, and the message does not have to be copied anywhere. The RETDAT time (ACReq) does depend upon the number of devices, as each device packet must be processed asthe request is initialized. The Update task time here is used to build the first reply and pass it to the network hardware. The last column shows the ongoing extra time needed in the Update task to build a reply each time it is due. The first number is the time when a reply is not due, in which a (short) time is needed to accumulate readings for computing an average value. The second number is the time to actually build a reply, which is of course similar to the time for building the first reply. The times in the Updt column and the Total column summarize this data. For example, the time needed to fully process reception of a 20-device (40 packets) nonserver request is about 2.5 ms, with about 1 ms needed to build each reply.

Here follows network activity related to the 20 device case, which uses 40 device packets. The request message is received and intialized, and a reply is built for a prompt first response. All device packets are local, so this is a very simple case. The turnaround time is 3 ms. Subsequent replies are delivered every two cycles at a time early in the 15Hz cycle.

```
Node Size   Ptr    HrMn Sc Cy ms
E0D1 02B0 R 163222 1514:04-05+34  Request
E0D1 00CA T 18CD4A 1514:04-05+37  First reply to requester
E0D1 00CA T 18CE2E 1514:04-07+ 2  Second reply
E0D1 00CA T 18CF12 1514:04-09+ 2  Third reply
```

Server request timing

Here are times for a **server** RETDAT request for 2-byte analog readings and settings to be delivered at 7.5Hz. To make this a simple example, all device packets refer to the same node that is not the node receiving the request.

#dev	*SNAP*	*ANet*	*ACReq*	*Updt*	*Total*	*SNAP*	*ANet*	*ACReq*	*Updt*	*∆Serv*
1	0.19	0.09	0.54	0.29	1.11	0.18	0.08	0.20	0.28	0.1
10	0.21	0.10	0.63	0.34	1.28	0.18	0.09	0.26	0.30	0.13
20	0.30	0.10	0.73	0.44	1.57	0.19	0.08	0.35	0.32	0.14
40	0.33	0.10	0.95	0.54	1.92	0.21	0.09	0.51	0.34	0.20
80	1.55	0.10	1.40	1.09	4.14	0.26	0.09	0.81	0.44	0.31

There are more steps for the server case, because the data must be retrieved from another node. The RETDAT time (ACReq) depends somewhat on the size of the request, but it only has to forward the request, as the request included none of its own devices in this test example. In summary, the time to initialize the server request for the case of 20 devices (40 packets) is about 1.6 ms. The time to process each reply from the contributing node and build the reply message for the requester is about 1.25 ms. These times are somewhat smaller than those for the nonserver case, as the server node is not doing the bulk of the work of actually generating the data.

Here follows the network activity for the 20-device (40 device packets) server case. For this simple test, only a single contributing node was used. When the request was received, it was forwarded to the other node, which initialized the request and delivered a prompt first reply to the server node, which in turn passed it on to the requester. The total turn-around at the server node for the first reply was 6 ms. Subsequent replies are delivered at 40 ms into the IRM's 15Hz cycle, allowing plenty of time to receive all reply fragments.

```
Node Size   Ptr    HrMn Sc Cy ms
E0D1 02B0 R 16B622 1445:01-07+46  Request
E131 02B0 T 18EA6C 1445:01-07+47  Forward request
E131 00CA R 16BC22 1445:01-07+51  First reply from (only) contributing node
E0D1 00CA T 18ED36 1445:01-07+52  to requester
E131 00CA R 16C222 1445:01-09+ 3  Second reply from contributing node
E0D1 00CA T 18EE1A 1445:01-09+40  to requester
```

Here follows the network activity related to the 80-device (160 device packets) case. Because of the large size of the request message datagram, it arrives in fragments which first have to be reassembled before the message can be formally received for processing. For this test, the requesting node was a test front end on token ring, which uses a 2KB MTU; this is why the request arrived in 3 fragments but was forwarded on ethernet in only two fragments. Because the reply message size is only about 700 bytes, no fragmentation is used for its delivery. In this example, the contributing node received the request late in the cycle, so its reply was slightly delayed due to updating its own data pool early in its cycle. Even so, the turn-around time at the server node was 20 ms as measured from the arrival time of the first request fragment to the delivery of the first reply.

```
Node Size   Ptr    HrMn Sc Cy ms
E0D0 0248 R 163822 1333:21-02+54  Fragment #1
E0D0 05D8 R 163E22 1333:21-02+57  #2
```

```
E0D0 0230 R 164422 1333:21-02+57  #3
E0D1 0A30 R 031854 1333:21-02+58  Accept reassembled datagram
E131 05D8 T 18335C 1333:21-02+60  Fragment #1 (Forward entire request)
E131 0468 T 183DA6 1333:21-02+61  #2
E131 02AA R 164A22 1333:21-03+ 6  First reply from contributing node
E0D1 02AA T 184228 1333:21-03+ 8  to requester
E131 02AA R 165622 1333:21-05+ 5  Second reply from contributing node
E0D1 02AA T 1844EC 1333:21-05+41  to requester
E131 02AA R 166222 1333:21-07+ 5  Third reply from contributing node
E0D1 02AA T 1847B0 1333:21-07+41  to requester
E131 02AA R 166E22 1333:21-09+ 5  etc.
```

Of course more complex examples of RETDAT timing can be measured, but the above should provide a picture of the real time operation of the support for the RETDAT protocol that is described in this note.